CS 341L Fall 2009 Lab 4

Assigned: Monday, 28 September 2009

Due: 11:59pm, Tuesday 6 October 2009 by attachment as an email to labturnin.cs341l@gmail.com

You should attach your program with your last name and the lab number in the name of the file, e.g., "knockel-lab4.S".

No late assignments will be accepted unless circumstances warrant an extension for the whole class (e.g., server is down all weekend, etc.). You are expected to do your own work as an individual effort, copying the code of others is considered cheating.

Note that this is a 10 point, one week lab, despite what the tentative schedule said and that the due date of 11:59pm on Tuesday is the exception for this lab, not the norm for future labs.

The purpose of this lab is to familiarize you with recursion and writing recursive functions in MIPS.

Overview:

recursion /rɪˈkɜrʒən/
*-**noun***
See *recursion*.

A recursive function is a function defined in terms of itself. A recursive function consists of two cases: a base case and a recursive case that reduces the problem to something closer to the base case.

Consider this recursive definition of factorial in C:

```
int factorial(int n) {
    if (n <= 1) {
        return 1; // base case
    } else {
        return n * factorial(n - 1); // reduce problem closer to base case
    }
}
```

Assignment:

In combinatorics, the Catalan number C_n is defined as follows:

$$C_n = \begin{cases} 1 & \text{if } n \leq 1 \\ \frac{2(2n-1)}{n+1} C_{n-1} & \text{otherwise} \end{cases}$$

Catalan numbers solve many counting problems, such as the number of expressions containing n correctly matched pairs of parentheses. For example, all possible expressions with three correctly matched parenthesis pairs are {*((())), ()(()), ()()(), (())(), (()())*}, so $C_3 = 5$.

In general; C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7... = 1, 1, 2, 5, 14, 42, 132, 429...

You are to implement a recursive catalan() function in handwritten MIPS assembly. Your MIPS function should implement catalan() according to the following C declaration:

```
unsigned int catalan( unsigned int n );
```

So a C program that calls catalan(3) should return 5.

No other functions except catalan() should be declared in your *.S turnin file, and you may not call other functions in your catalan() function except catalan().

No testing suite will be provided with this lab. If you wish to test your catalan() function (strongly recommended), then you may write your own test code in a separate file in a language of your choice that can link with your assembled *.o file (e.g. in C, C++, MIPS assembly). This file should implement main() and perform a variety of tests of catalan() of your invention. You may use the testing suite provided with Lab 3 as a reference on how to set this up if you are a bit rusty with object linking or the other compilation stages, especially if you want to write your own Makefile to facilitate this. Note that, if you choose to write tests, then you should still only turn in your *.S file with your handwritten MIPS implementation of catalan()—writing tests is strictly for your benefit!

Assembly:

gcc -Wall -Wextra knockel-lab4.S -c -o knockel-lab4.o

Example compilation and linking to test file written in C (optional):

gcc -Wall -Wextra main.c -c -o main.o

gcc -Wall -Wextra main.o knockel-lab4.o -o lab4